03

Errores en conexión

Transcripción

[00:00] Y ven, lo prometido es deuda. Y como les dije, les voy a enseñar una forma de forzar la ejecución de este método haya o no haya error en una estructura de try/catch. ¿Y cómo hago esto? Existe un tercer elemento en la estructura de try/catch, una palabra reservada y se llama finally. También tiene su propio alcance su propio scope.

[00:35] ¿Y final y básicamente, qué es lo que va a decir? De una u otra forma vas a cerrar la conexión. Entonces, si yo borro aquí y borro aquí, fíjense que yo no estoy ya cerrando conexión en ningún lado, excepto en finally. ¿Por qué? Porque finally sí o sí. allá éxito o haya error, él va a ejecutar ese método. Vamos a guardarlo aquí.

[01:05] Vamos a ponerle para eso un Sysout, y vamos a darle aquí "Ejecutando finally". Perfecto. Entonces aquí vamos a probar cuándo estamos lanzando el error. Le damos play y perfecto, abrió conexión, recibió los datos, dio excepción, ejecutó finally y cierra la conexión. Vamos aquí a nuestra clase conexión, volvemos a comentar para probar el otro escenario, ejecutamos nuevamente, abriendo datos, recibiendo, ejecutando finally, cerrando conexión.

[01:47] ¿Entonces con esto qué aseguramos? Todos los casos con finally vamos a cerrar siempre la conexión. Esa es la tercera estructura del try/catch. ¿Entonces qué hemos aprendido hasta este momento?

[02:04] Try/catch es una estructura muy, muy poderosa para manejar errores y también vamos a asegurarnos de que en casos como este que necesitemos cerrar la conexión, entonces con finally tenemos el soporte necesario para que, ya sea que encontremos un error o que no encontremos ningún error, finally va a ejecutarse.

[02:30] Ahora, otra cosa que me gustaría que se lleven también de estos videos, no solamente es bueno hacer pruebas con el camino feliz, si es que no hay errores. Recuerden siempre probar el camino lanzando la excepción y también el camino sin lanzar la excepción. Puede ser un trabajo un poco manual pero nos aseguramos de que nuestro código está protegido ante los dos casos que hemos mapeado hasta el momento.

[02:59] Entonces esa práctica es muy buena para que puedan escribir código escalable y seguro, código fuerte que no va a tener errores no mapeados o por lo menos va a considerar la mayoría de casos para mapear esos errores y tratarlos ya sea en una estructura try/catch o de alguna otra forma que ya ustedes pueden idear ahí.

[03:26] Bueno, ya les presenté el finally. Podemos tener finally en otro lugar aquí también pero ya eso lo vamos a ver en el siguiente video. Nos vemos.